Harri Leino

Ilmaisen tunkeutumisenestojärjestelmän teho väistöhyökkäyksiä vastaan

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

16.5.2017

Metropolia

Tekijä Otsikko Sivumäärä Aika	Harri Leino Ilmaisen tunkeutumisenestojärjestelmän teho väistöhyökkäyksiä vastaan 28 sivua 16.5.2017
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikka
Suuntautumisvaihtoehto	Tietoverkot
Ohjaaja	Lehtori Erik Pätynen

Väistötekniikat (evasion techniques) ovat menetelmiä, joilla verkkoliikennettä muokataan tarkoituksena naamioida uhkia tietoturvajärjestelmiltä. Insinöörityössä selvitettiin, kuinka hyvin ilmainen palomuuriohjelma kykenee havaitsemaan ja torjumaan tällä tavalla piilotettuja uhkia ilman, että käyttäjän täytyy osata säätää palomuurin sääntöjä. Lisäksi selvitettiin, millaisin tavoin käyttöjärjestelmät ovat alttiita tällaisille tekniikoille ja millaisin menetelmin verkkoliikenteen datapaketteja voidaan muokata tietoturvaohjelmien kiertämiseksi.

Työhön valittiin yksi palomuuriohjelma ja väistötekniikoita automatisoiva testityökalu ja pystytettiin virtuaaliympäristö, jossa haitallista verkkoliikennettä ohjattiin palomuurin läpi. Ensimmäisessä testissä kokeiltiin, torjuuko palomuuri kaikki väistötekniikat yksittäin käytettynä. Toisessa testissä suoritettiin 20 000 automatisoitua hyökkäystä, joissa käytettiin satunnaisesti kahdesta neljään eri väistötekniikoiden yhdistelmää. Ensimmäisessä testissä kaikki hyökkäykset estettiin, ja toisessa hyökkäyksistä estettiin 99,31 prosenttia.

Tuloksien pohjalta tehtiin johtopäätös, että ilmainen palomuuri on tehokas turva väistöhyökkäyksiä vastaan ja että yksinkertaisilla asetusmuutoksilla loputkin hyökkäyksistä saadaan todennäköisesti torjuttua. Se on siis varteenotettava vaihtoehto kaupallisille tuotteille.

Avainsanat	IDS, IPS, väistötekniikat, tietoturva

Author	Harri Leino
Title	Effectiveness of a free intrusion prevention system against evasion techniques
Number of Pages	28 pages
Date	16 May 2017
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Computer Networks
Instructor	Erik Pätynen, Senior Lecturer

Evasion techniques are methods of modifying network traffic in an attempt to disguise threats from information security devices. The objective of this thesis was to find out, whether or not free firewall software could detect and block threats disguised with these methods, without the user having to modify rules for the firewall. In addition, this thesis investigated how operating systems are vulnerable to these methods, as well as how data packets could be modified in order to circumvent security software.

One firewall software and automated evasion tool was chosen for testing, and a virtual environment was set up, where harmful network traffic was sent through the firewall. The first test was to try which evasion techniques worked when used alone. In the second test, 20,000 automated attacks were run with random combinations of two to four evasions. The firewall successfully blocked everything in the first test, and 99.31 percent in the second.

It was concluded that a free firewall is an efficient way of combating evasion techniques, and that with small rule modifications, the rest of the attacks could probably be blocked as well. The free firewall is therefore a considerable alternative for commercial products.

Keywords	IDS, IPS, evasion techniques, information security

Sisällys

Lyhenteet

DAQ	*Data Acquisition*. Snort-ohjelman paketinkaappausmoduuli.
DPI	*Deep Packet Inspection*. Palomuurin tapa tarkastaa koko datapaketti.
ECN	*Explicit Congestion Notification*. IP-protokollan ruuhkanhallintatyökalu.
HIDS	*Host-based IDS*. Isäntälaitteen kautta toimiva IDS.
HTTP	*Hypertext Transfer Protocol*. Tiedonsiirtoprotokolla.
IETF	*The Internet Engineering Task Force*. Internetin standardeja luova järjestö.
IDS	*Intrusion Detection System*. Ohjelma tai laite, joka passiivisesti etsii sisäverkon liikenteestä uhkia.
IP	*Internet Protocol*. Tiedonsiirtoprotokolla.
IPS	*Intrusion Prevention System*. Ohjelma tai laite, joka aktiivisesti torjuu sisäverkon liikenteestä uhkia.
ISO	*International Organization for Standardization*. Standardisointijärjestö.
MAC	*Media Access Control*. OSI-mallin toisen tason osa.
NGFW	*Next-Gen Firewall*. Nykyaikainen palomuuri, joka sisältää lukuisia tietoturvaominaisuuksia.
NIDS	*Network-based IDS*. Verkkolaitteessa oleva IDS.
OS	*Operating System*. Käyttöjärjestelmä.
OSI	*Open Systems Interconnect*. Malli, jolla luokitellaan verkkoliikennettä.
PAWS	*Protection Against Wrapping Sequence*. TCP-protokollan algoritmi.

pcap	*packet capture*. Tiedostotyyppi.
RFC	*Request For Comments*. IETF-järjestön asiakirja.
SYN	*Synchronize*. TCP-protokollan kontrollibitti.
TCP	*Transport Control Protocol*. Tiedonsiirtoprotokolla.
TSER	*Timestamp Echo Reply*. TCP-protokollan aikaleima-arvo.
TSval	*Timestamp value*. TCP-protokollan aikaleima-arvo.
UDP	*User Datagram Protocol*. Tiedonsiirtoprotokolla.
URG	*Urgent*. TCP-protokollan kontrollibitti.
URL	*Uniform Resource Locator*. Verkkoresurssin osoite.

1 Johdanto

Nykypäivänä yksi kehittyneiden palomuurien suurimmista haasteista ovat väistötekniikat. Niiden tarkoitus on naamioida verkkoliikennettä niin, että uhkat jäävät huomaamatta tietoturvalaitteilta. Jopa tehokkaat kaupalliset ratkaisut, uuden sukupolven palomuurit (engl. next-gen firewall, NGFW), ovat silloin tällöin kompuroineet testeissä [1]. Tämä on herättänyt miettimään vanhaa kysymystä: Onko kaupallinen ohjelmisto parempi kuin avoimen lähdekoodin kilpailija?

Tässä työssä on tarkoituksena tutkia, voidaanko ilmaisella modernilla palomuurilla torjua väistöhyökkäyksiä ilman, että käyttäjä joutuu itse laatimaan torjuntasääntöjä, eli toisin sanoen tuoreella asennuksella. Näin toivotaan saatavan selville, voidaanko esimerkiksi kotiverkkoon turvallisesti asentaa vähällä ylläpidolla ja tietotaidolla ilmainen tunkeutumisenestojärjestelmä (engl. Intrusion Prevention System, IPS) niin, että se huomaa ja torjuu salakavalat väistöhyökkäykset.

Minulla on hyvä yleinen tietopohja tietotekniikasta ja jonkin verran tietoverkkojen ja -liikenteen tuntemusta, mutta IPS- tai IDS (Intrusion Detection System) -ohjelmat ja väistötekniikat ovat uusi aihealue. Tämä huomioon ottaen ja työkuorman rajoittamiseksi tässä työssä valitaan vain yksi palomuuriohjelma, joka pystytetään virtuaaliympäristöön ja johon käytetään väistötekniikoihin erikoistunutta testaustyökalua. Näin pyritään selvittämään, onko nykypäivän hakkerille täysin mitätön haaste ohittaa yritykset nostaa kodin tai pienyrityksen tietoturvatasoa.

2 Tietoliikenne lyhyesti

Insinöörityössä käytettävät tekniikat vaativat jonkinasteista verkkoliikenteen tuntemusta, ja tämän työn lukijan odotetaan ymmärtävän vähintään IP-osoitteiden ja porttien toimintaa. Väistöhyökkäykset kuitenkin käyttävät toiminnassaan tietoliikenteen pakettien yksityiskohtaista muokkaamista, joten näiltä osin on hyvä avata tietoliikenteen perusteita ja tiedonsiirtoprotokollien toimintaperiaatteita. Tämä myös mahdollistaa työn myöhemmissä vaiheissa tietoliikenneaiheisten lyhenteiden luontevan käytön, mikä helpottaa työn lukemista.

2.1 OSI-malli

OSI (Open Systems Interconnect) -malli on ISO (International Organization for Standardization) -standardisointijärjestön luoma konsepti, jonka avulla luokitellaan tietoliikenteen eri osia loogisille tasoille. Seuraavaksi kuvaan pintapuolisesti OSI-mallin kerrokset ja mitä osia tietoliikenteestä niille sijoittuu:

1 Fyysinen kerros (Physical)

- media, jonka kautta sähköinen signaali kulkee lähettäjän ja vastaanottajan välillä, kuten Ethernet-kaapeli, sarjakaapeli tai langaton signaali
- siirtomuoto, eli kulkeeko signaali analogisesti vai digitaalisesti, miten yksittäisiä bittejä ja bittijonoja tulkitaan, ja niin edelleen

2 Siirtoyhteyskerros (Data Link)

- fyysisen bittivirran jakaminen datapaketeiksi ja bittivirheiden korjaus
- Ethernet-kehykset ja tiedonsiirto paikallisessa aliverkossa
- käsittää MAC (Media Access Control) -osoitteet

3 Verkkokerros (Network)

- liikenteen reititys sisäverkossa ja verkkojen välillä
- käsittää IP (Internet Protocol) -osoitteistuksen
- IP-pakettien paloittelu pienempiin osiin eli fragmentteihin

4 Kuljetuskerros (Transport)

- pyrkii varmistamaan, että kahden pisteen välinen pakettivirta pääsee perille kokonaisena ja oikeassa järjestyksessä
- tunnetuimpia protokollia TCP (Transmission Control Protocol) ja UDP (User Datagram Protocol)
- TCP-pakettien paloittelu pienempiin osiin eli segmentteihin
- liikenteen ohjaus portteihin

5 Istuntokerros (Session)

- huolehtii kahden päätylaitteen välisen keskusteluyhteyden aloittamisesta, ylläpitämisestä ja lopetuksesta

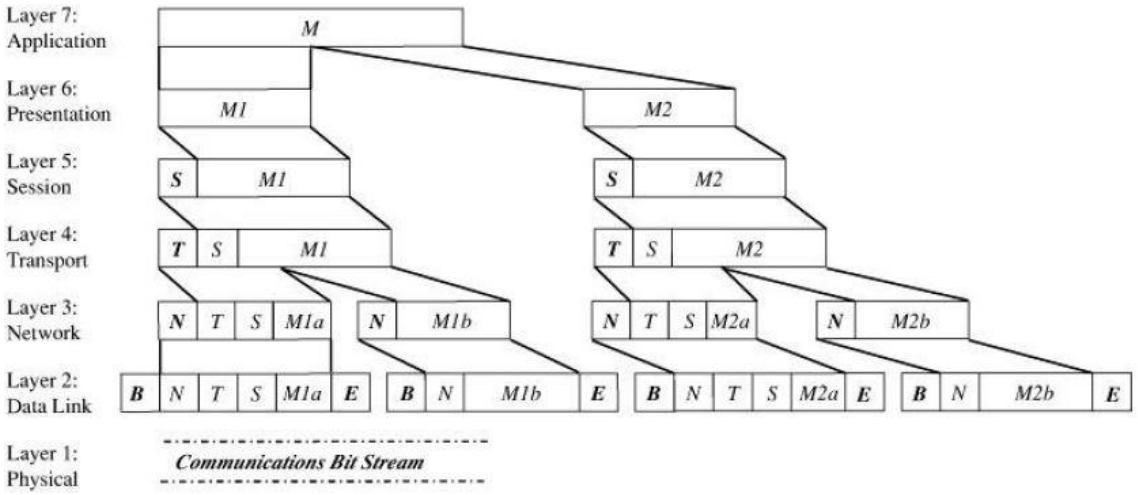
6 Esitystapakerros (Presentation)

- muotoilee dataa ihmisen silmälle sopivaksi, eli toimii ikään kuin tulkkina ihmisen ja sovelluskerroksen datan välillä

7 Sovelluskerros (Application)

- käsittää varsinaisen datan, jota verkossa halutaan kuljettaa, siinä missä alemmat kerrokset liittyvät sen kuljetukseen.

Jokainen alempi kerros koteloi ylemmän kerroksen sisällön, kuten kuvassa 1 havainnollistetaan. Toisen kerroksen Ethernet-kehys (engl. frame) sisältää kolmannen kerroksen IP-paketin, joka taas sisältää neljännen kerroksen TCP-paketin ja niin edelleen.

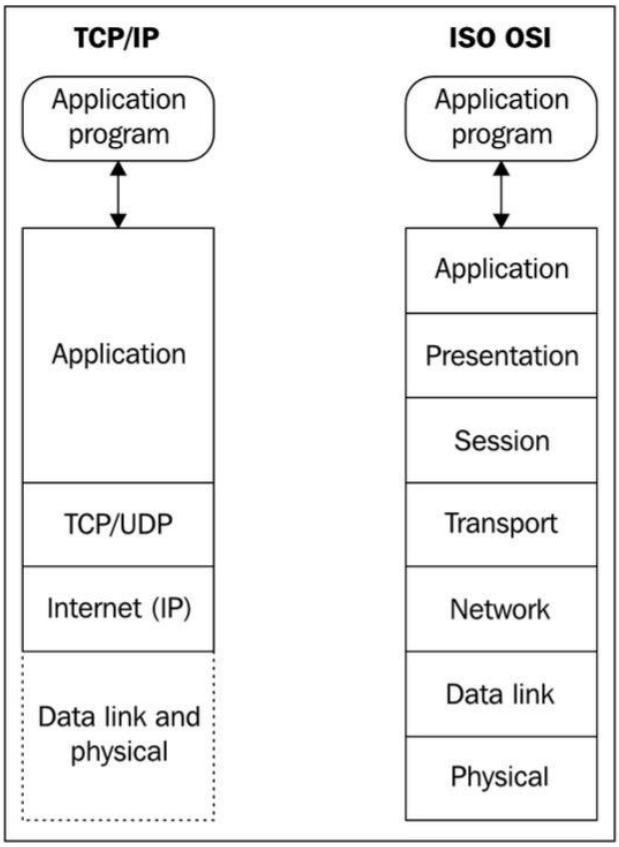


Kuva 1. Viestin M pakkaaminen datapaketeiksi [2, s. 424].

Datapakettien kulkiessa verkossa jokainen reititin tavallisesti tutkii otsikosta kohteen IP-osoitteen, jotta se saa selville, minne paketti kuuluu toimittaa eteenpäin, ja lähettää sen sitten edelleen.

2.2 TCP/IP-pino

OSI-mallin ohella käytetään toista mallia, TCP/IP, joka on nimetty sen kahden tärkeimmän protokollan mukaan. Kuten kuvasta 2 näkyy, molemmat mallit kuvailevat samaa toimintaa hieman eri tavoin, käyttäen suurilta osin samoja protokollia.



Kuva 2. TCP/IP- ja OSI-mallin eroavaisuudet [3, s. 7].

TCP/IP eroaa OSI-mallista niin, että siinä istunto-, esitystapa- ja sovelluskerros yhdistetään yhdeksi sovelluskerrokseksi, eikä se pitkään käsittänyt OSI-mallin alinta tasoa ollenkaan. OSI-mallia saatetaankin kuvailla turhan monimutkaiseksi, ja sen täysi sisällyttäminen verkkoliikenteeseen hidastaisi sen toimintaa [2, s. 424]. Siinä missä OSI-malli on lähinnä abstrakti konsepti, TCP/IP on ennemmin tapa niputtaa toimivia protokollia yhdeksi perheeksi. Kun laitekehittäjä toteuttaa tuotteeseensa tuen TCP/IP-protokollapinolle, se kykenee käsittelemään olennaista verkkoliikennettä. Molempia malleja alettiin kehittää 1970-luvun lopulla, kun haluttiin yhtenäistää eri valmistajien erilaiset tavat toteuttaa verkkoliikennettä [4].

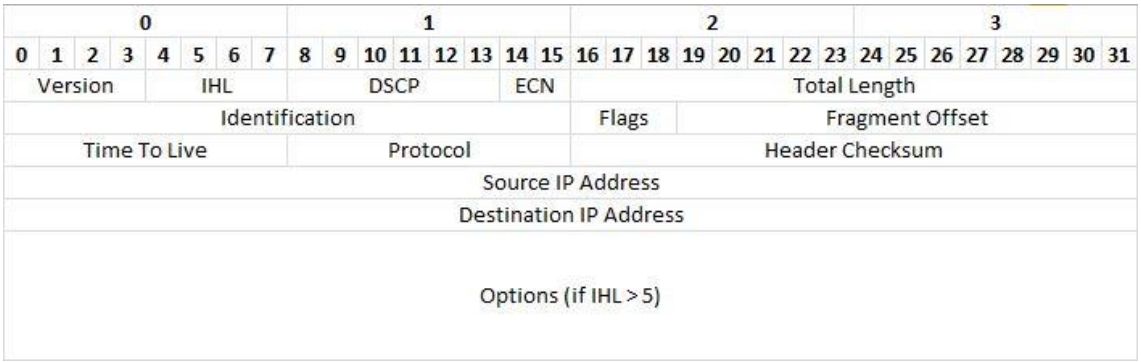
2.3 Tiedonsiirtoprotokollia

Valtaosan internetiä kannattavista yksittäisistä verkkoprotokollista on kehittänyt IETF (The Internet Engineering Task Force) -järjestö, jonka tehtävänä on luoda RFC (Request for Comments) -asiakirjoja. Osa näistä asiakirjoista määrittelee standardeja sille, kuinka tiettyjä protokollia kuuluu toteuttaa verkossa toimiviin laitteisiin.

Väistötekniikoita tämä koskettaa siltä kannalta, että verkkoon yhdistäviin laitteisiin ja ohjelmiin ei aina ohjelmoida TCP/IP-pinon toteutusta samalla tavalla. RFC-standardeihin tehdään muutoksia ajan myötä, ja niissä voi olla virheitä. Esimerkiksi alkuperäisen standardin mukaan TCP-paketin otsikon kiireellisen datan osoittimen pitäisi osoittaa kiireellisen datan viimeisen tavun jälkeiseen tavuun [5, s. 17], mutta myöhemmässä standardissa sen ilmoitetaan olevan virheellinen ilmaisu ja osoittimen kuuluisikin osoittaa kiireellisen datan viimeiseen tavuun [6, s. 84]. Yli kaksi vuosikymmentä näiden julkaisujen jälkeen on selvitetty, että lähes jokainen testattu TCP/IP-toteutus käyttää tätä ensimmäistä väärää tapaa, ja kehotetaan luopumaan kokonaan TCP:n kiireellisen datan käytöstä [7, s. 5]. Seuraavaksi kuvailen joitakin IP- ja TCP-protokollan ominaisuuksia, jotka liittyvät suoraan väistötekniikoiden toimintaan.

IPv4-protokolla

IPv4-väistötekniikoiden kirjo on kapea, koska IP-paketin otsikko on suhteellisen yksinkertainen toiminnaltaan verrattuna esimerkiksi TCP-paketin vastaavaan, kuten kuvista 3 ja 4 voidaan havaita.



Kuva 3. IP-paketin otsikko.

Fragmentointiin kuitenkin liittyy paljon hyväksikäyttöpotentiaalia, koska palomuurilaitteet eivät aina osaa koota pakettivirtoja samalla tavalla kuin suojeltavat laitteet. Fragmentteja saa standardin mukaan koota yhteen vain, jos protokollakenttä on niissä sama [8, s. 29], ja tätä on todistetusti voitu käyttää hyväksi ainakin yhdessä tapauksessa [9]. Fragmentoitujen virtojen aikakatkaisua voidaan myös potentiaalisesti käyttää hyväksi, sillä on olemassa vain suositus sille, kuinka kauan vastaanottava laite jää odottamaan yhteen kuuluvia fragmentteja [8, s. 27]. Jos palomuurilaite käyttää 30 sekunnin aikakatkaisua ja palvelin 60 sekunnin ja hyökkääjä pitää 45 sekunnin tauon keskellä pakettivirtaa, palomuuri voi mahdollisesti jäädä käsittelemään uutta puolikasta pakettivirtaa samalla, kun palvelin kokoaa pakettivirran yhteen. Lisäksi yksittäisten pakettien elinaikaa eli TTL (Time To Live) -arvoa, jota paketin matkatessa jokainen reititin vähentää yhdellä, voidaan muokata sellaiseksi, että paketti saavuttaa palomuurin muttei päätylaitetta. Tätä hyväksikäyttäen laitteet voidaan harhauttaa kokoamaan toisistaan poikkeavat datakokonaisuudet [10].

TCP-protokolla

Koska TCP on tilallinen protokolla, sen hallinnoiminen on erittäin monimutkaista verrattuna esimerkiksi tilattomaan UDP:hen, jonka otsikko sisältää vain neljä yksinkertaista kenttää. Kuten kuva 4 näyttää, TCP käyttää useita kolmikirjaimisia hallintabittejä, mikä lisää potentiaalia häiritä sen toimintaa aivan eri tavalla kuin esimerkiksi yksi isompi osoitetta kuvaava kenttä.

<table>
<tr><th colspan="8">0</th><th colspan="8">1</th><th colspan="8">2</th><th colspan="8">3</th></tr>
<tr><th>0</th><th>1</th><th>2</th><th>3</th><th>4</th><th>5</th><th>6</th><th>7</th><th>8</th><th>9</th><th>10</th><th>11</th><th>12</th><th>13</th><th>14</th><th>15</th><th>16</th><th>17</th><th>18</th><th>19</th><th>20</th><th>21</th><th>22</th><th>23</th><th>24</th><th>25</th><th>26</th><th>27</th><th>28</th><th>29</th><th>30</th><th>31</th></tr>
<tr><td colspan="16">Source port</td><td colspan="16">Destination port</td></tr>
<tr><td colspan="32">Sequence number</td></tr>
<tr><td colspan="32">Acknowledgment number (if ACK set)</td></tr>
<tr><td colspan="4">Data offset</td><td colspan="3">Reserved
000</td><td>N
S</td><td>C
W
R</td><td>E
C
E</td><td>U
R
G</td><td>A
C
K</td><td>P
S
H</td><td>R
S
T</td><td>S
Y
N</td><td>F
I
N</td><td colspan="16">Window Size</td></tr>
<tr><td colspan="16">Checksum</td><td colspan="16">Urgent pointer (if URG set)</td></tr>
<tr><td colspan="32">Options (if data offset > 5. Padded at the end with "0" bytes if necessary.)</td></tr>
</table>

Kuva 4. TCP-paketin otsikko.

TCP-segmentteihin pätevät samat ongelmat kuin IP-fragmentteihin, mutta segmenttien käsittelyä voidaan vaikeuttaa lisää muokkailemalla kuittausnumeroita (engl. acknowledgement number), järjestysnumeroita (engl. sequence number), ikkunan kokoa (engl.

window size) ja otsikon kokoa (engl. data offset). Segmentteihin voidaan luoda päällekkäisyyksiä, ja koska missään ei määritellä, kumman data kahdesta päällekkäisestä segmentistä priorisoidaan, molempia tapoja käytetään vaihdellen järjestelmästä toiseen. Tämä mahdollistaa haitallisen datan piilottamisen segmenttien väliin.

Tavallisen liikenteen mukaan voidaan lisätä sinne kuulumattomia satunnaisia SYN (Synchronize)- ja URG (Urgent) -bittejä hämäämään palomuuria, ja yhteyksiä voidaan TIME-WAIT-säännön vastaisesti aukaista juuri suljettuun porttiin. Aikaleima- ja aikaleimakaikuarvoissa (engl. timestamp value, TSval ja timestamp echo reply, TSER) voidaan käyttää mielivaltaisia arvoja, mikä saa PAWS (Protect Against Wrapped Sequence) -algoritmia noudattamattoman päätylaitteen hyväksymään paketin virheellisellä aikaleimalla, siinä missä standardien mukainen palomuuri saattaa jättää sen huomiotta.

3 Palomuurin toimintaperiaate

Palomuuri on perinteisesti ollut laite, joka asetetaan paikallisen verkon ulkorajalle suojelemaan esimerkiksi kodin tai yrityksen sisäverkkoa ulkomaailman tietoturvauhilta. Käytännössä palomuuriksi voidaan mieltää mikä tahansa ohjelma tai laite, joka tietoturvan nimissä suodattaa verkkojen välistä tietoliikennettä soveltaen sisäisiä sääntöjään. Yleensä palomuureille määritelläänkin ulkopuolen ja sisäpuolen rajapinnat vastaavasti epäluotetuksi ja luotetuksi alueeksi. Tavallisesti ulkopuolelta estetään lähes kaikki liikenne sisäänpäin, kun taas sisäpuolelta ulospäin lähes kaikki sallitaan. Sisäverkosta ulkomaailmaan liikenne on tavallisesti melko vapaata, kun taas ulkopuolelta sisäverkkoon ei pääse lähes mikään paketti ilman aloitetta sisäpuolelta. [2, s. 509.]

Yksinkertaisimmillaan palomuuri on ohjelma, jonka läpi kaikki paikallisen verkon ja ulkomaailman välinen tietoliikenne ohjataan ja joka sitten vertaa jokaista datapakettia listaan sääntöjä ja päättää sen perusteella, päästääkö se paketin läpi vai pudottaako sen. Palomuureja voidaan yleensä konfiguroida kummin päin tahansa – pudottaa kaikki liikenne, joka ei osu mihinkään listan säännöistä (engl. default deny, oletuksena kielletään), tai sallia kaikki liikenne, johon mikään säännöistä ei kohdistu (engl. default permit, oletuksena sallitaan). Tietoturva-asiantuntijat suosivat vahvasti edeltävää vaihtoehtoa [2, s. 509].

Yksinkertaisin paketteja suodattava (engl. packet filtering) palomuuri tarkastaa datapaketeista vain kolme kohtaa: lähteen IP-osoite ja portti, kohteen IP-osoite ja portti ja käytettävä kuljetusprotokolla (esimerkiksi TCP) [2, s. 511]. Tämä riittää, kun halutaan esimerkiksi sallia ulkoverkosta vain HTTP (Hypertext Transfer Protocol) -liikenne tietylle web-palvelimelle, jolloin vain porttia 80 käyttävä liikenne sallitaan, mikä on hyödyllistä esimerkiksi porttiskannauksen kaltaisia tiedusteluhyökkäyksiä vastaan. Tällainen palomuuri ei kuitenkaan suojele sisäverkkoa näiden sallittujen protokollien väärinkäytöltä, kuten esimerkiksi SYN flood -palvelunestohyökkäyksiltä tai muuten haitalliselta liikenteeltä web-palvelimen ja ulkoisen hyökkääjän välillä.

Paketteja tarkastava (engl. packet inspecting) palomuuri puolestaan kykenee pitämään kirjaa verkkoliikenteen datavirroista sen sijaan, että se päästäisi läpi kaiken sääntöihin sopivan liikenteen pakettikohtaisesti. Toisin sanoen tilallisten protokollien kuten TCP:n tilaa on mahdollista seurata. Näin voidaan varmistaa, että vain niin kutsuttua laillista liikennettä kulkee sisäverkkoon. Suodattavan ja tarkastavan palomuurin paketinkäsittelytapoja kutsutaan myös vastaavasti tilattomaksi (engl. stateless) ja tilalliseksi (engl. stateful) tarkastamiseksi. [2, s. 512.]

Loogisesti seuraava konsepti on tunkeutumisenhavaitsemisjärjestelmä eli IDS ja tunkeutumisenestojärjestelmä eli IPS. IDS:n ja IPS:n oleellinen ero on, että IDS vain tutkii paketteja, jotka sille lähetetään kopiona sisäverkon liikenteestä, kun taas IPS aktiivisesti tutkii ja estää haitallisen liikenteen pääsyn sisäverkkoon (tai esimerkiksi tärkeän tiedon pääsyn ulos). Monesti kuitenkin ainoa ero IDS:n ja IPS:n välillä on sen aktiivisuus – seuraako se vain passiivisesti liikennettä vai muokkaako se sitä aktiivisesti. [2, s. 519.]

IDS voi toimia kahdella eri periaatteella: tunnisteisiin perustuva ja poikkeamapohjainen (engl. anomaly-based) [2, s. 520]. Tunnisteisiin perustuvalla IDS:llä on käytettävänään laaja tietokanta erilaisia malleja eli tunnisteita, joiden perusteella verkkoliikenteen seasta voidaan tunnistaa uhkia. Poikkeamapohjainen IDS taas muodostaa verkkoliikenteen pohjalta mallin tavallisesta liikenteestä ja antaa hälytyksiä, kun tästä mallista poikkeavaa liikennettä syntyy. On tietenkin mahdollista käyttää molempia periaatteita samassa järjestelmässä. IDS voidaan myös luokitella sijaintinsa mukaan; HIDS eli isäntäpohjainen (engl. host-based) IDS tarkkailee yksittäistä tietokonetta virustorjuntaohjelman tavoin, kun taas NIDS eli verkkopohjainen (engl. network-based) IDS on verkon rajamailla suojelemassa koko sisäverkkoa erillisessä laitteessa. [11, s. 19.]

Ideaalitilanteessa esimerkiksi yritysverkon laidalla onkin ensin suodattava palomuuri tai tarkastava palomuuri – mahdollisesti molemmat samassa laitteessa – joka suodattaa suurimman osan muun muassa yleisistä palvelunestohyökkäyksistä, ja sen takana vasta IDS [12]. Pakettien tarkastus on laskentatehollisesti hyvin raskasta, ja kuten luvussa 4.3 mainitaan, IPS:n uhkantorjuntakykyyn vaikuttaa olennaisesti sen resurssien käyttöaste.

Raja paketteja tarkastavan palomuurin ja IPS:n määritelmän välillä on häilyvä, ellei olematon. Jotkut vetävät rajan pakettien tarkastuksen tasoon; palomuuri tarkastaa vain OSI-mallin 4. tasoon saakka, eli se katsoo esimerkiksi IP- ja TCP-pakettien otsikoita, muttei niiden sisällä kulkevaa dataa, kun taas IPS tarkastaa koko paketin. Jälkimmäistä tapaa kutsutaan pakettien syvätarkastukseksi (engl. deep packet inspection, DPI). Jotkut taas erottelevat tehon mukaan, esimerkiksi Juniperin SSG-sarjan laitteiden DPI-ominaisuus eroaa SRX-sarjan IPS-ominaisuudesta vain tunnisteiden määrässä ja sitä kautta laitteiden tehovaatimuksessa [13]. Myöhemmin tässä työssä viitataan palomuuriin, IDS:ään ja IPS:ään pelkästään palomuurina lukemisen yksinkertaistamiseksi.

4 Väistöhyökkäykset

Väistöhyökkäyksiksi kutsun tässä työssä haitallista verkkoliikennettä, johon on sovellettu erilaisia väistötekniikoita (tai kiertomenetelmiä, engl. evasion techniques). Väistötekniikat ovat tapoja, joilla pyritään muokkaamaan verkkoliikennettä sellaiseksi, että onnistutaan harhauttamaan palomuuri päästämään läpi haitallista liikennettä, jonka se muuten estäisi.

4.1 TCP/IP-pinon toteutuserot eri järjestelmissä

Osa väistötekniikoista pyrkii hyväksikäyttämään eroavaisuuksia siinä, kuinka palomuuri ja kohdelaite käsittelevät erilaisia paketteja. Pohjimmainen ongelma on, kuten aiemmassa osiossa on mainittu, että kaikissa käyttöjärjestelmissä ja palomuureissa TCP/IP-pinoa ei ole toteutettu täysin samalla tavalla. Verkossa sijaitsevasta laitteesta onkin mahdollista päätellä paljastavia yksityiskohtia jo pelkästään sen perusteella, millaisia TTL-alkuarvoja se laittaa IP-paketteihin [14]. Hyökkääjän on siis mahdollista räätälöidä haitallista liikennettä, mikäli hän ymmärtää, kuinka uhrilaitteen tapa käsitellä paketteja eroaa muista laitteista tai yleiskäytännöstä ja erityisesti sitä suojelevasta palomuurista.

Tästä hyvänä esimerkkinä toimii aiemmin mainittu TCP-protokollan kiireellisen osoittimen toiminta. Aiemman kuvailun mukaisesti riippuen siitä, kumpaa käsittelytapaa lähettävä ja vastaanottava laite käyttää, tiedonsiirrossa voi tapahtua odottamattomia datahäviöitä, joita voidaan käyttää hyväksi hyökkäyksissä [15]. Jos verkossa lähetetään viesti `HIGHLIGHT`, mutta se paloitellaan kolmeen TCP-segmenttiin `HIG`, `HLI` ja `GHT` ja merkitään keskimmäinen paketti kiireelliseksi, palomuuri saattaa koota paketin oikein yhteen, kun taas kohdelaite saattaa pudottaa yhden merkin pois kiireellisen segmentin lopusta niin, että kootussa viesti lukee `HIGHLGHT`. Täten jos palomuuri etsii liikenteestä tunnistetta `HIGHLIGHT`, sitä voidaan kiertää lähettämällä TCP-segmentit `HIG`, `HLIX` ja `GHT`, jolloin palomuuri kokoaa viestin `HIGHLIXGHT`, joka ei laukaise hälytystä, kun taas kohdelaite kokoaa viestin `HIGHLIGHT`.

Yksittäisiä IP- ja TCP-väistötekniikoita kuvataan luvussa 5.2.

4.2 Hämäännyttäminen

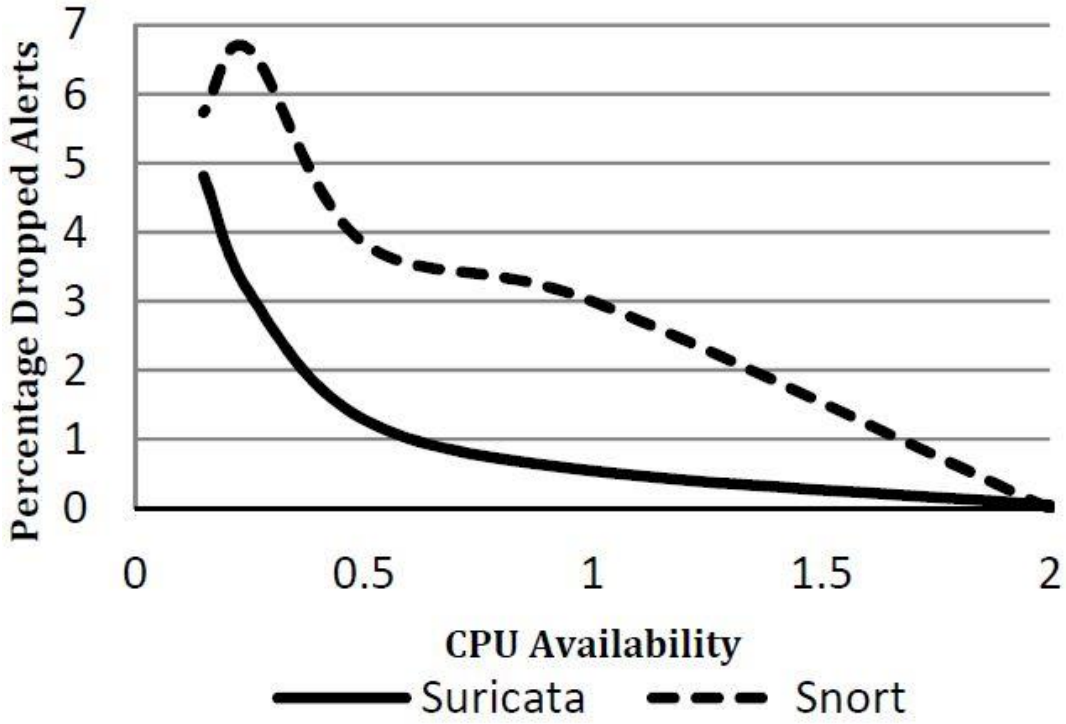
Toisenlainen väistötekniikoiden ryhmä on hämäännyttäminen (engl. obfuscation), jota kuva 5 havainnollistaa. Näillä menetelmillä hyökkäyksen varsinaista haitallista, selkokielistä koodiosuutta muokataan sellaiseen muotoon, että vastaanottava laite osaa tulkita sitä oikein, mutta se ei laukaise palomuurin tunnistepohjaista hälytystä [16]. Esimerkiksi URL (Uniform Resource Locator) -kentässä lähetettävä selkoteksti `/etc/passwd` voidaan koodata heksadesimaaliseen muotoon `%2f%65%74%63%2f%70%61%73%73%77%64`. Kohdelaitteen HTTP-palvelin osaa kääntää koodin takaisin selkokieleksi, mutta palomuurilla ei välttämättä tällaista ohjeistusta ole.

```
GET /phpBB2/viewtopic.php?t=1&highlight=%2527.passthru(passthru(chr(101).chr
(99).chr(104).chr(111).chr(32).chr(112).chr(114).chr(101).chr(100).chr(97).chr
(116).chr(111).chr(114).chr(52).chr(95).chr(98).chr(101).chr(103).chr(105).chr
(110).chr(59).chr(32).chr(99).chr(100).chr(32).chr(47).chr(101).chr(116).chr
(99).chr(59).chr(32).chr(99).chr(97).chr(116).chr(32).chr(112).chr(97).chr(115).chr
(115).chr(119).chr(100).chr(59).chr(32).chr(101).chr(99).chr(104).chr(111).chr
(32).chr(112).chr(114).chr(101).chr(100).chr(97).chr(116).chr(111).chr(114).chr
(52).chr(95).chr(101).chr(110).chr(100))).%2527 HTTP/1.1

GET /phpBB2/viewtopic.php?
t=1&dX=ZWNobyBwcmVkYXRvcjRfYmVnaW47IGNkIC9ldGM7IGNhdCBwYXNzd2Q7IGVjaG8gcHJlZGF0&AQp=
%24swB%3q%24_TRG;%24iZl%3q%24UGGC_TRG_INEF;CNffgUeH(onfR64_qrPBQR(EnJHEyqrPBqR(%
24swB[qK].%24swB[NDc])));&highlight=pMM%252%37.EVA%25%36c(R%%361WU%7%32LdeC%%36fDe
(%%373t%25%352_r%6ft%253%313(%2524_GE%25%35%34[%2578dR%%35d)))|%%327IqCvo HTTP/1.1

GET /%70%68%70%42%42%32/%76%69%65%77%74%6f%70%69%63%2e%70%68%70?%74=%31&%49%6c%44=%
5a%57%4e%6f%62%79%42%77%63%6d%56%6b%59%58%52%76%63%6a%52%66%59%6d%56%6e%61%57%34%
37%49%47%4e%6b%49%43%39%6c%64%47%4d%37%49%47%4e%68%64%43%42%77%59%58%4e%7a%64%32%
51%37%49%47%56%6a%61%47%38%67%63%48%4a%6c%5a%47%46%30%62%33%49%30%58%32%56%75%5a%
41%41%41&%63%70=%24%52%4b%%33%71%24%5f%54%52%47;%24%6a%6e%%33%71%24%55%47%47%43%5f%
54%52%47%5f%49%4e%45%46;%43%4e%46%66%67%55%65%68%28%6f%6e%46%72%36%34%5f%51%52%70%
62%71%72%28%65%6e%6a%48%65%79%51%52%70%62%51%72%28%24%6a%6e%5b%56%79%51%5d%29%29%
29;&%68%69%67%68%6c%69%67%68%74=%75%71%4c%4d%6d%77%67%69%68%%32%37%7c%65%56%41%6c%
28%52%25%36%31%57%75%52%4c%64%45%63%4f%25%34%34%65%28%53%25%37%34%52%5f%72%25%36%
66%54%31%%33%33%28$%%35%66%47%45%54%5b%63%70%%35%64%29%29%%32%39,%%32%37%45%4b%4a%
4a%6d%53%6b%56 HTTP/1.1
```

Kuva 5. Sama HTTP-pyyntö ensin selkotekstinä, sitten hämäännytettynä ja sen jälkeen vielä URL-koodattuna.

4.3 Palvelunesto

Väistötekniikoiksi voidaan myös mieltää palvelunestohyökkäykset, jotka kohdistuvat suoraan tai epäsuorasti palomuurin resursseihin. On mahdollista lähettää verkkoon massiivista turhaa liikennettä, jonka kohdelaite jättää täysin huomiotta, mutta palomuuri kuluttaa kaiken tehonsa sen tutkimiseen [17]. Palomuurin käyttökuormasta riippuu, kuinka suurella todennäköisyydellä se huomaa haitallista liikennettä. Eräässä tutkimuksessa on todettu, kuten kuva 6 havainnollistaa, että palomuurin kyky havaita uhkia vähenee prosessorikuorman lisääntyessä [18].



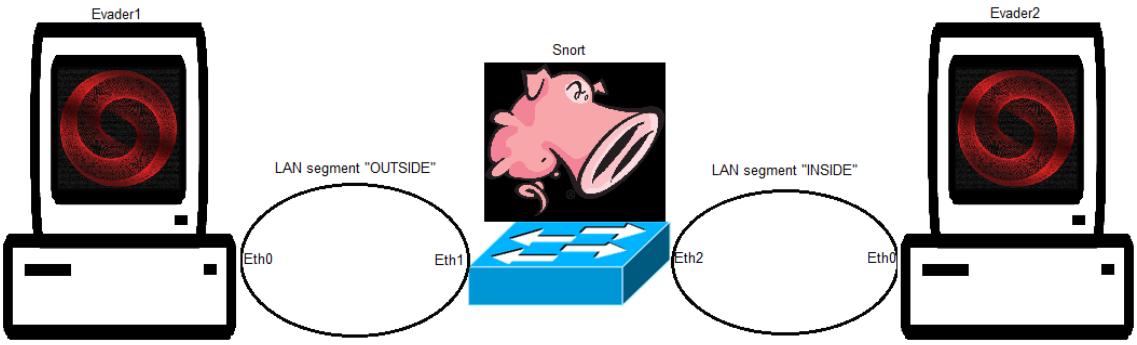
Kuva 6. Snort ja kilpailija Suricata: kun prosessorien käytettävyys (availability) laskee, ohi päässeiden haittojen (dropped alerts) määrä nousee [18].

Tässä työssä ei kuitenkaan käsitellä hyökkäyksiä, jotka kohdistuvat suoraan tai epäsuorasti palomuurin resursseihin.

5 Työympäristö ja käytetyt ohjelmat

5.1 Topologia

Insinöörityön testaukset suoritetaan virtuaaliympäristössä VMware Playerilla. Työssä käytetään kolmea virtuaalilaitetta: hyökkääjälaite, uhrilaite ja näiden välissä palomuurilaite, kuten kuvassa 7 havainnollistetaan. Sekä hyökkääjä- että uhrilaitteella on Evader-käyttöjärjestelmä, ja palomuurilaitteella on alustana Debian Linux ja sen päällä Snort. Ohjelmia käsitellään lisää myöhemmin.



Kuva 7. Virtuaaliympäristön topologia.

Kaikki laitteet ovat samassa virtuaaliverkossa 192.168.10.0/24, mutta niiden verkkosovittimet on asetettu kuvan 7 mukaisiin verkkosegmentteihin.

5.2 Evader-ohjelma

Palomuurin testaamiseen on tässä työssä valittu väistöhyökkäyksiin erikoistunut Evader-ohjelma. Sen kehitti alun perin suomalainen palomuurivalmistaja Stonesoft, ja se on ollut saatavilla ilmaiseksi verkosta. Helmikuussa 2014 Stonesoft siirtyi McAfeen omistukseen, ja vuoden 2016 alkupuolella Stonesoft sulautettiin uuteen Forcepoint-yhtiöön ja Evader poistui yleisestä jakelusta. Tässä työssä käytetty Evaderin versio on ladattu McAfeen verkkosivuilta maaliskuussa 2015, ja on mahdollista, että siihen on lisätty ominaisuuksia latauksen ja jakelusta poiston välillä.

Evaderin toiminta perustuu siihen, että hyökkääjälaitteella käytetään verkon kautta palomuurin takana sijaitsevaa uhrikonetta vastaan vanhaa tunnettua haavoittuvuutta, joka jokaisen nykyaikaisen palomuurin pitäisi tunnistaa. Hyökkäyksen luomaa verkkoliikennettä sitten muokataan yhdellä tai useammalla erilaisella väistötekniikalla tavoitteena, että palomuuri päästää erehdyksessä liikenteen ohitseen. Evader myös oletuksena hämäännyttää pakettien haitallisen dataosan.

Evaderissa on mahdollista valita kolmen haavoittuvuuden välillä: Conficker-madon käyttämä RPC-protokollan haavoittuvuus Microsoft Serverissä, keskustelupalstasovellus phpBB:n haavoittuvuus tai Windows Remote Desktop (RDP) -protokollan palvelunestohyökkäyksen mahdollistava haavoittuvuus. Haavoittuvuuden valinta vaikuttaa myös siihen, millaisiin protokolliin kohdistuvia väistötekniikoita voidaan käyttää. Näistä ensimmäiseen voidaan soveltaa IP-, TCP-, NetBIOS-, SMB- ja MSRPC-protokolliin

liittyviä väistötekniikoita. Toiseen taas voidaan soveltaa IP-, TCP- ja HTTP-väistöjä ja kolmanteen vain IP- ja TCP-väistöjä. Seuraavaksi listaan Evaderin käyttämät IP- ja TCP-protokolliin kohdistuvat väistötekniikat.

IPv4-protokolla

- Fragmentaatio: Datavirta paloitellaan moneksi erilliseksi IP-paketiksi.
- Fragmenttien järjestys: IP-pakettien virta lähetetään joko päinvastaisessa järjestyksessä, satunnaisessa järjestyksessä, viimeinen paketti ensimmäisenä tai ensimmäinen paketti viimeisenä.
- Pakettien kopiointi: Lähetetään aidon liikenteen rinnalla samannäköistä liikennettä, mutta niin, että sen haitallisen koodin osuutta on sekoitettu yhdellä muutamasta menetelmästä.

TCP-protokolla

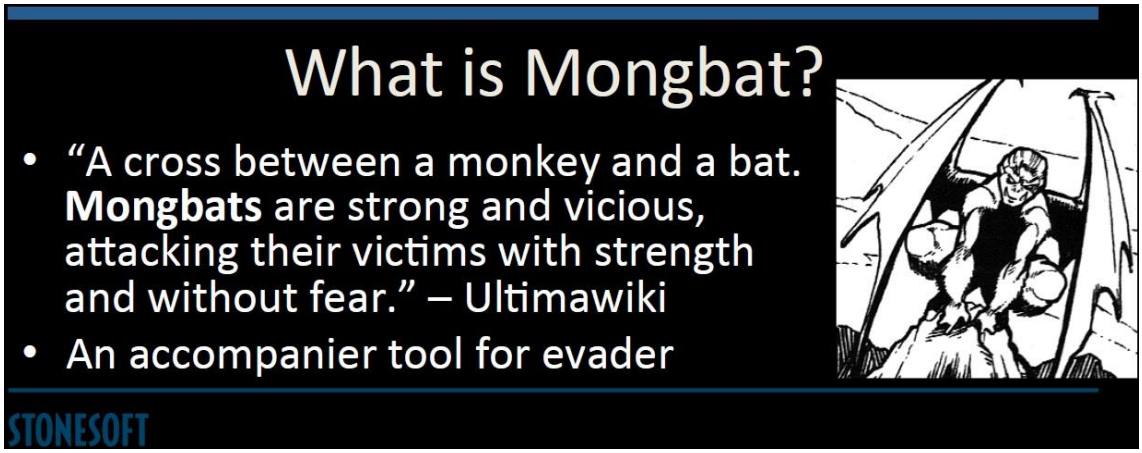
- Häirintäsilppu: TCP-segmenttien ohella lähetetään tekaistuja segmenttejä, joiden otsikossa on joko huono tarkistussumma, NULL-bitti, järjestys- tai kuittausnumero ikkunan ulkopuolella, liian pieni tai suuri otsikon koko tai URG-bitti.
- Järjestysnumero: Ensimmäinen järjestysnumero asetetaan manuaalisesti.
- Ruuhkanhallinta: Ei käytetä ruuhkanhallintaan tarkoitettuja otsikon bittejä.
- Segmentaatio: Asetetaan lähetettäville TCP-paketeille mielivaltainen enimmäiskoko tai satunnainen enimmäiskoko kahden luvun välillä.
- Segmenttien järjestys: TCP-segmenttien järjestystä muutetaan, kuten ylempänä IP-pakettien kanssa.
- Segmenttien limittäisyys: Lähetetään liikenteessä TCP-paketteja, joiden järjestysnumeroissa on päällekkäisyyksiä, ja valitaan sijoitetaanko haitallinen koodi ennen vai jälkeen tulevaan pakettiin – on käyttöjärjestelmäriippuvaista, ylikirjoitetaanko TCP-pakettivirtaa kootessa ennen tullut data jälkeen tulleella vai toisin päin.
- OS spoofing: Muokataan pakettien vapaammin valittavia arvoja kuten TCP-ikkunan kokoa sen näköiseksi, että liikenne näyttäisi tulevan tietystä käyttöjärjestelmästä.
- PAWS: Lähetetään hyökkäysliikenteen ohella kopioitu pakettivirta, jossa aikaleimoja on muokattu taaksepäin toivoen, että palomuuri tarkastaa vain kopioidun virran ja ohittaa oikean virran virheellisenä.
- Ikkunan koko: Säädetään manuaalisesti TCP-ikkunan kokoa.

- SYN-bitti: Lähetetään ensimmäisen SYN-paketin mukana myös varsinainen dataliikenne tai lisätään SYN-bitti ensimmäiseen dataliikenteen pakettiin.
- TIME-WAIT: Avataan TCP-yhteys ja suljetaan se, ja standardeista poiketen avataan saman tien samaan TCP-porttiin uusi yhteys odottamatta, että TIME-WAIT-aika ensin umpeutuu.
- TSER-muokkaus: Lähetetään TCP-paketeissa virheellisiä aikaleimoja.
- URG-bitti: Lähetetään osa liikenteestä kiireellisenä.

IP- ja TCP-protokollan lisäksi myöhemmässä testivaiheessa käytetään myös HTTP-protokollaan perustuvia väistötekniikoita, mutta niitä ei käsitellä sen syvemmin. Valtaosa maailmanlaajuisen verkon tiedonsiirrosta tapahtuu IP- ja TCP-protokollaa hyödyntäen, siinä missä HTTP-väistöjä voidaan käyttää vain web-palvelimien kanssa.

5.2.1 Mongbat-ohjelma

Evaderin mukana tulee väistöhyökkäysten automatisointiin tarkoitettu ohjelma Mongbat, joka on kuvan 8 mukaisesti saanut nimensä Ultima-pelimaailmassa elävältä apinan ja lepakon risteytykseltä [19].



Kuva 8. Mongbat-ohjelman nimen selitys [19].

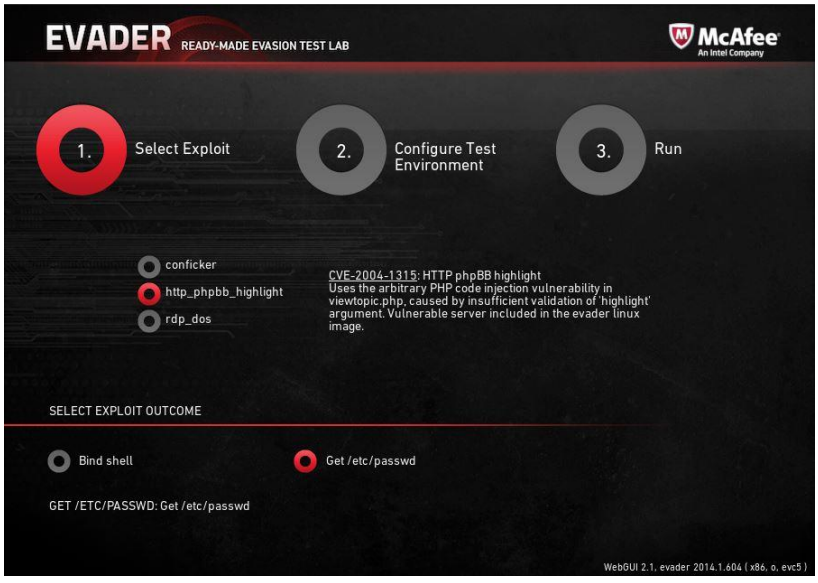
Mongbatilla on mahdollista suorittaa samanaikaisesti useita Evader-instansseja, jotka kaikki tekevät väistöhyökkäyksiä ennalta määritellyllä haavoittuvuudella ja satunnaisilla väistötekniikoilla ja raportoivat takaisin Mongbat-instanssille. Jokaisen Evader-ajon yhteydessä suoritetaan ensin hyökkäys ilman varsinaista haitallista koodiosuutta `--payload=clean` -argumentilla, minkä jälkeen suoritetaan samoilla parametreilla toi-

nen hyökkäys, joka sisältää haitallisen osion. Näin pyritään sulkemaan pois mahdolliset väärät positiiviset tulokset, eli tapahtumat, joissa joko yhteys ei toimi uhri- ja hyökkääjälaitteen välillä tai palomuuri torjuu liian innokkaasti vaaratonta liikennettä. Suorittamisen jälkeen Mongbat antaa raportin, jossa on tilastotietoa hyökkäyksistä ja lista komennoista, joilla kyetään suorittamaan onnistuneet hyökkäykset samoilla parametreilla.

5.2.2 Asennus ja käyttö

Evaderin ladattava levykuva on oikeastaan itsenäisesti toimiva Ubuntu Linuxin pohjalta rakennettu käyttöjärjestelmä, ja se voidaan joko asentaa tietokoneelle tai käynnistää live-levynä. Asennuksen hyvänä puolena on mahdollisuus tallentaa verkkoasetuksia. Käyttöjärjestelmässä on Evader, Mongbat, Wireshark-pakettianalysaattori ja lukuisia perushyödyllisiä työkaluja. Evaderia voidaan käyttää joko komentoriviltä tai graafisesta käyttöliittymästä. Käyttöjärjestelmää voidaan käyttää myös uhrilaitteena, mikäli käytetään phpBB-haavoittuvuutta, koska siitä voidaan käynnistää web-palvelin, joka sisältää vanhan paikkaamattoman version phpBB-keskustelupalstasta. Järjestelmän käynnistyessä ruudulle avautuu tekstitiedosto, jossa on ohjeet kunkin ohjelman käynnistykseen.

Graafinen käyttöliittymä jakautuu kolmeen välisivuun. Kuva 9 näyttää näistä ensimmäisen, jolla valitaan haavoittuvuus ja haettu lopputulos. Toisella säädetään verkkoasetukset kuntoon ja kolmannella valitaan haavoittuvuuden kanssa käytettävät väistötekniikat.



Kuva 9. Hyökkäyksessä käytettävän haavoittuvuuden valinta.

Kuvan 10 havainnollistamassa kolmannessa ruudussa voidaan myös Automatic evasions -valinnalla suorittaa Mongbat ainakin teoriassa, mutta suoritettavassa komennossa on työssä käytetyssä Evader-versiossa jostain syystä argumentti `--workers=0`, mikä tarkoittaa, että Mongbat ei käynnistä yhtäkään Evader-instanssia.

Kuva 10. Esimerkki onnistuneesta hyökkäyksestä kahta väistötekniikkaa käyttäen.

Hyökkäyksissä käytetty komento näkyy kuitenkin aina lokissa, josta sen voi kopioida ja suorittaa muokattuna komentoriviltä, samoin kuin Evader-komennot manuaalisissa hyökkäyksissä. Lokista voidaan myös ladata pcap-tiedosto, josta voidaan katsella hyökkäyksen pakettivirtaa Wireshark-työkalulla.

5.3 Snort-tunkeutumisenestojärjestelmä

Työn laajuuden rajaamisen takia testattavaksi valittiin vain yksi palomuuri. Valinnan aikaan ehdokkaita olivat Snort, Bro ja Suricata. Bro oli vielä hyvin uusi, pienen tiimin ajama projekti, jolla oli hyvin vähän näkyvyyttä, joten se päätettiin jättää huomiotta. Suricata oli myös uusi mutta hyvin tuettu palomuuri, ja testeissä se on pärjännyt jok-

seenkin samalla tasolla Snortin kanssa [17]. Snort kuitenkin herätti enemmän luottamusta pitkällä historiallaan, laajalla dokumentaatiollaan ja näkyvyydellään. Työhön harkittiin myös pfSense-ohjelmaa, mutta se toimi Snortin pohjalta, joten koettiin paremmaksi käyttää suoraan Snortia ilman välikäsiä.

Snort on avoimen lähdekoodin tunkeutumisenestojärjestelmä, joka kykenee reaaliaikaiseen liikenneanalyysiin ja pakettien kirjaamiseen [20]. Sitä kehitetään Linux-ympäristöön, vaikka siitä on olemassa myös Windowsissa toimiva Winsnort-versio. Snortille voidaan ladata sen verkkosivuilta sääntöjä. Yksi sääntö sisältää tunnisteen ja ohjeen, mitä tehdä, kun se aktivoidaan. Sääntöjä on kolmessa eri ryhmässä: yhteisösäännöt, rekisteröityjen käyttäjien säännöt tai tilaussäännöt. Yhteisösäännöt ovat avoimen yhteisön yhdessä ylläpitämiä sääntöjä ja kaikille vapaasti ladattavissa, kun taas rekisteröityjen ja tilaavien asiakkaiden säännöt sisältävät huomattavasti laajemman sääntökokonaisuuden. Jälkimmäisten olennainen ero on, että rekisteröityneet asiakkaat saavat säännöt käyttöönsä 30 päivää tilaavia asiakkaita myöhemmin. Sääntöpakettien verrannollisesta laajuudesta kertoo se, että rekisteröityjen sääntöpaketin koko on 50-kertainen yhteisösääntöihin verrattuna. Snortin verkkosivuilta voidaan myös ladata lukuisia Snortiin liittyviä sivuohjelmia, kuten Pulled Pork, joka lataa automaattisesti uusia sääntöjä, tai BASE, joka mahdollistaa Snortin hälytysten analysoimisen graafisella verkkoliittymällä.

5.3.1 Toiminta

Snort poimii verkkoliikenteestä datapaketteja DAQ-tiedonkeruumoduulin (engl. data acquisition) kautta, joka on yhteydessä libpcap-paketinkaappauskirjaston kanssa, joka puolestaan on yhteydessä suoraan verkkorajapintaan. DAQ toimiikin eräänlaisena tulkkina libpcap-kirjaston ja Snortin välillä, ja DAQ-asennuspaketissa on muutama eri moduuli erilaisiin toimintaympäristöihin. Esimerkiksi oletusmoduuli afpacket muodostaa käytettäessä Ethernet-sillan kahden verkkosovittimen välille ja kykenee muokkaamaan verkkoliikennettä vain kytkevässä laitteessa, kun taas yhdessä iptables-ohjelman kanssa toimiva moduuli NFQ mahdollistaa verkkoliikenteen muokkaamisen reitittävässä laitteessa.

Snort voi toimia IDS:nä, IPS:nä tai pelkkänä pakettikaapparina, ja sen läpi voidaan myös viedä pcap-paketinkaappaustiedostoja.

5.3.2 Esikäsittelijät ja säännöt

Snort sisältää lukuisia modulaarisia esikäsittelijöitä, joiden tehtävä on muun muassa torjua väistötekniikoita. Ajettaessa Snort käsittelee pakettivirrat ensin esikäsittelijöillä, ennen kuin niitä verrataan varsinaisten tunnistesääntöjen kanssa. IPS-moodissa aivan ensimmäisenä liikenne viedään maine-esikäsittelijän kautta, joka tarkastaa, hylätäänkö paketit suorilta käsin tai lähetetäänkö ne käsittelemättöminä palomuurin läpi, riippuen käyttäjän määrittelemistä listoista. Tämän jälkeen, OSI-mallin järjestystä myötäillen, Frag3 kokoaa ensin fragmentoidut IP-paketit kokoon, sen jälkeen Stream5 kokoaa segmentoidut TCP-paketit yhtenäiseksi kokonaisuudeksi, ja sen jälkeen liikenne vielä normalisoidaan, ennen kuin käytetään sovellustason esikäsittelijöitä. Normalisoinnissa muun muassa poistetaan TCP-paketeista ECN (Explicit Congestion Notification) -bitit ja hylätään paketit, joita ei voida koota oikein osaksi TCP-virtaa.

Näiden esikäsittelijöiden jälkeen pakettivirta viedään oletuksena vielä runsaan kymmenen eri sovellustason protokollaan erikoistuneen esikäsittelijän läpi riippuen paketin sisällöstä, ennen kuin niihin sovelletaan tunnistesääntöjä. Tässä kohtaa prosessia useimmat tunnetut väistötekniikat on poistettu pakettivirrasta, ja esikäsittelijät ovatkin erikoistuneet väistötekniikoiden torjumiseen. Varsinaisilla tunnistesäännöillä ei täten ole lisättävää prosessiin väistötekniikoiden kannalta.

5.3.3 Asennus ja käyttö

Helpoiten Snortin saa asennettua suoraan paketinhallintatyökalulla, mikäli Snort on saatavilla valmiina pakettina valittuun käyttöjärjestelmään. Se voi tosin vaikeuttaa ohjelman eri osien löytämistä käyttöjärjestelmän uumenista, eikä se salli muun muassa DAQ-moduulien lisäystä, ja täten Snort onkin suositeltavaa asentaa manuaalisesti. Tämän luvun menetelmät on kohdennettu Debian-käyttöjärjestelmälle, kun Snort asennetaan toimimaan IPS-moodissa.

Vähimmillään Snort vaatii toimiakseen libpcap-, libdnet- ja DAQ-paketit. Libdnetin saa yleensä asennettua paketinhallintatyökalulla. Jotkin säännöt saattavat vaatia pcre3-kirjaston voidakseen käyttää säännöllisiä lausekkeita, joten sekin on hyvä asentaa. Jotkin verkkosovittimet taas kokoavat itse pakettivirtoja, mikä voi vaikuttaa Snortin toimintaan, joten on suositeltavaa asentaa myös ethtool-ohjelma, jolla nämä verkkosovittimen asetukset saadaan helposti pois käytöstä.

```
apt-get -y install ethtool libdnet-dev libdumbnet-dev
libpcre3-dev
```

Seuraavat ethtool-komennot kannattaa automatisoida, jottei niitä tarvitse suorittaa manuaalisesti jokaisen käynnistyksen jälkeen. Debianin tapauksessa lisätään tiedostoon /etc/rc.local ennen riviä `exit0` jokaista verkkorajapintaa kohden:

```
ethtool --offload eth1 rx off tx off

ethtool -K eth1 gso off

ethtool -K eth1 gro off
```

Seuraavaksi ladataan ja asennetaan libpcap ja DAQ:

```
cd /usr/src

wget http://www.tcpdump.org/release/libpcap-1.8.1.tar.gz

wget  http://sourceforge.net/projects/snort/files/snort/daq-
2.0.6.tar.gz

tar -zxf libpcap-1.8.1.tar.gz

tar -zxf daq-2.0.6.tar.gz

cd libpcap-1.8.1

./configure --prefix=/usr && make && make install

cd ../daq-2.0.6

./configure && make && make install
```

Tämän jälkeen on hyvä päivittää jaettujen kirjastojen lista:

```
echo >> /etc/ld.so.conf /usr/lib

echo >> /etc/ld.so.conf /usr/local/lib && ldconfig
```

Sitten asennetaan Snort:

```
cd ..
```

```
wget
http://sourceforge.net/projects/snort/files/snort/snort-
2.9.9.0.tar.gz

tar -zxf snort-2.9.9.0.tar.gz

cd snort-2.9.9.0

./configure --enable-sourcefire && make && make install
```

Tämän jälkeen luodaan Snortille tarvittavat hakemistot, tiedostot, käyttäjä ja oikeudet:

```
mkdir /usr/local/etc/snort

mkdir /usr/local/etc/snort/rules

mkdir /var/log/snort

mkdir /usr/local/lib/snort_dynamicrules

touch /usr/local/etc/snort/rules/white_list.rules

touch /usr/local/etc/snort/rules/black_list.rules

groupadd snort && useradd -g snort snort

chown snort:Snort /var/log/snort

cp ./etc/*.conf* /usr/local/etc/snort

cp ./etc/*.map /usr/local/etc/snort

mkdir snortrules

cd snortrules
```

Tämän jälkeen Snortin sivuilta ladataan sääntöpaketti, jota varten kannattaa hankkia rekisteröidyt tunnukset. Helpoiten lataus tapahtuu selaimella, sillä komentoriviltä sääntöjä haettaessa voi tulla ongelmia käyttäjiä yksilöivän oinkcoden kanssa. Latauksen jälkeen säännöt siirretään Snortin kansioihin:

```
tar -zxf snortrules-snapshot-2990.tar.gz

cp -R preproc_rules /usr/local/etc/snort/
```

```
cp -R rules /usr/local/etc/snort/

cp -R so_rules /usr/local/etc/snort/

cp -R ./etc/* /usr/local/etc/snort/
```

Sitten konfiguroidaan Snort:

```
vim /usr/local/etc/snort/snort.conf
```

- Riviltä 45 kannattaa muuttaa avainsana `any` oman kotiverkon osoitteeksi, esimerkiksi 192.168.3.0/24 ja seuraavan rivin vastaava muotoon `!$HOME_NET` . Tämä ei ole pakollista, mutta se tarkentaa sääntöjen kohdistusta ja täten vähentää Snortin kuormaa.
- Rivin 104 seudulta täytyy varmistaa, että kaikki `var *_PATH` -rivit osoittavat oikeaan kansioon.
- Rivin 365 kohdalta kannattaa varmistaa, että `preprocessor` -rivit eivät ole kommentoituja.
- Asetus `config policy_mode:inline` täytyy sijoittaa johonkin kohtaan tiedostoa, loogisesti riville 188.
- Rivillä 159 asetetaan DAQ-moodi seuraavilla asetuksilla:
- `config daq: afpacket`
- `config daq_dir: /usr/local/lib/daq`
- `config daq_mode: inline`
- `config daq_var: buffer_size_mb=1024`
- Kannattaa myös varmistaa rivistä 542 eteenpäin, että käytössä ovat kaikki sääntötiedostot, joita halutaan käyttää.

Nyt voidaan tallentaa muutokset ja sulkea tiedosto.

Snort käynnistetään komentoriviltä seuraavalla komennolla:

```
snort -A console -u snort -g snort -c
/usr/local/etc/snort/snort.conf -i eth1:eth2 -Q
```

Kahva `-A` kertoo, kirjoitetaanko hälytykset lokitiedostoon kokonaisina (full), nopeassa muodossa (fast) vai suoraan ruudulle (console). Kahvat `-u` ja `-g` kertovat, minä käyttäjänä ohjelma ajetaan. Tietoturvan kannalta on huono käytäntö ajaa Snort kaikilla oike-

uksilla. Kahva `-c` määrittelee käytettävän konfiguraatiotiedoston, ja `-i` määrittää kuunneltavan verkkosovittimen tai verkkosovitinparin. Kahva `-Q` puolestaan käskee Snortin toimimaan inline-moodissa, eli se antaa luvan muokata liikennettä.

Oletusarvoisesti suurin osa valmiista tunnistesäännöistä on hälytysmoodissa (engl. alert), mikä tarkoittaa, että hälytyksen lisäksi paketeille ei tehdä mitään. Sääntöjä voidaan vapaasti muokata esimerkiksi pudotettavaksi (engl. drop) mikä tarkoittaa, että pakettia ei lähetetä eteenpäin hälytyksen tullessa. Suuri osa valmissäännöistä on oletuksena kommentoitu, eli niitä ei oteta huomioon ajossa.

6 Väistöhyökkäysten testaus

6.1 Testausmenetelmät

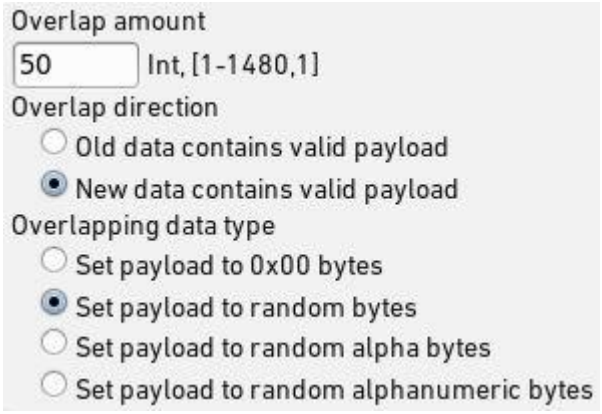
Tässä työssä haluttiin nähdä, kuinka hyvin Snort käsittelee väistöhyökkäyksiä ilman, että käyttäjältä vaaditaan erityistä tunnistesääntöjen tuntemusta. Niinpä Snortin valmiiden sääntöjen joukosta etsittiin ja aktivoitiin vain kolme hieman toisistaan eroavaa sääntöä, jotka liittyvät Evaderin käyttämään phpBB-haavoittuvuuteen ja näyttävät jotakuinkin tältä:

```
drop tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
(msg:"SERVER-WEBAPP phpBB viewtopic double URL encoding at-
tempt"; flow:to_server,established; con-
tent:"viewtopic.php"; content:"highlight="; distance:0;
content:"%25"; distance:0; reference:cve,2004-1315;
classtype:web-application-attack; sid:33293; rev:1;)
```

Jokainen näistä kolmesta säännöstä etsii HTTP-datasta avainsanoja `viewtopic.php`, `highlight=` ja `%25`. Jälkimmäinen koodattu merkki tulkataan web-palvelimella puolikkaaksi heittomerkiksi ', jota käytetään injektiohyökkäyksessä. Yksikin sääntö olisi riittänyt, mutta haluttiin varmuuden vuoksi asettaa jokainen drop-moodiin.

Kun Snort oli käynnissä, uhrilaite-Evaderilla käynnistettiin phpBB-palvelin, minkä jälkeen hyökkääjälaitteelta valittiin phpBB-haavoittuvuus ja haitalliseksi kuormaksi etäkomentotulkin avaus. Tämän jälkeen hyökkääjälaitteella alettiin kokeilla järjestelmällisesti jokaista yksittäistä väistöliikettä kaikin mahdollisin asetuksin.

Laaja-alaisista numeromuuttujista, kuten kuvan 11 esimerkistä, ei kokeiltu jokaista mahdollista vaihtoehtoa. Skaalan alkupäästä, loppupäästä ja keskivaiheilta kokeiltiin kuitenkin muutamaa eri numeroa. Tekniikoita pyrittiin myös kohdentamaan niin, että esimerkiksi TCP-paketit paloiteltiin juuri palomuurin etsimän avainsanan kohdalta.



Kuva 11. Yhden väistötekniikan toteutuksessa voi olla useita eri vaihtoehtoja.

Tämän jälkeen suoritettiin Mongbatilla automatisoitu hyökkäys, jossa käytettiin samaa haavoittuvuutta ja haitallista kuormaa, ja yhdisteltiin satunnaisesti kahdesta neljään eri väistötekniikkaa samanaikaisesti neljällä eri Evader-instanssilla. Ohjelman annettiin olla käynnissä noin 20 000 hyökkäyksen ajan.

6.2 Tulokset

Snort onnistui torjumaan jokaisen yksittäistä väistötekniikkaa käyttävän hyökkäyksen kaikilla asetuksilla. Kuva 12 havainnollistaa, kuinka Mongbat suoritti ensin ilman haitallista kuormaa yhteensä 20 289 yhteydenottoa, joista 45 epäonnistui. Näiden jälkeen se suoritti yhteensä 20 244 hyökkäystä, joista 139 pääsi palomuurin lävitse. Satunnaisista hyökkäyksistä siis 0,69 prosenttia onnistui. Onnistuneista hyökkäyksistä muutaman toimivuus varmistettiin vielä manuaalisesti.

	Total	Clean	Exploit
Attempts	20289	20289	20244
Failures	20150	45	20105
Success	139	20244	139

Kuva 12. Mongbat-ajon tulokset.

Onnistuneita hyökkäyksiä selaillessa huomattiin trendi, jossa suurin osa onnistuneista hyökkäyksistä hyödynsi pakettien jakamista joko erittäin pieniin TCP-segmentteihin tai IP-fragmentteihin. Lisäksi huomattiin, että Snort kyllä nosti kyseisistä hyökkäyksistä hälytyksen, mutta vasta kun hyökkäävä laite lähetti TCP-yhteyden sulkevan RST-viestin. Tästä kiinnostuneena ja hieman työn parametreista poiketen päätettiin laittaa drop-moodiin vielä kaksi esikäsittelijöiden sääntöä, jotka estävät liikenteen, kun pakettivirrassa on epäilyttävän monta pienikokoista segmenttiä tai fragmenttia. Tämän jälkeen suoritettiin uudelleen pienemmän skaalan Mongbat-testi 10 000 hyökkäyksellä, jolloin hyökkäyksiä pääsi läpi 0,22 prosenttia, eli alle kolmasosa aiemmasta. Tällä työllä ei haluta helpottaa palomuurien kiertämistä, joten tässä kohdassa ei sen enempää tarkenneta, minkätyyppiset hyökkäykset olivat erityisen tehokkaita – vaikka ne saataisiinkin torjuttua yksinkertaisilla sääntömuutoksilla.

Toisin sanoen Snort onnistui torjumaan 99,31 prosenttia satunnaisista väistöhyökkäyksistä ilman sääntömuutoksia. Tuloksista voidaan tehdä johtopäätös, että Snort on erittäin pätevä poimimaan sekoitetun liikenteen seasta haitallista koodia.

7 Yhteenveto

Insinöörityössä lähdettiin selvittämään, riittääkö moderni ilmainen palomuuri torjumaan moderneja verkkouhkia. Testitulokset näyttäisivät osoittavan, että tämä tavoite toteutuu; ilman sääntömuutoksia kaikki yksittäiset väistötekniikat onnistuttiin torjumaan, kuten myös yli 99 prosenttia satunnaisista yhdistelmätekniikoista. Palomuuri myös antoi hälytyksiä epäilyttävästä liikenteestä jokaisen hyökkäyksen kohdalla, joka varmistettiin manuaalisesti satunnaisajon jälkeen. Tämä tarkoittaa, että pienillä sääntömuutoksilla tämän prosentin voi nostaa ehkä jopa sataan. Tämä tarkoittaa, että yrityksen ei välttämättä tarvitse käyttää omaisuutta tietoturvan hankintaan ja ylläpitoon, ainakaan väistötekniikoiden osalta.

Voitaisiin väittää, että yksityishenkilö on turvassa monimutkaisilta kohdennetuilta väistöhyökkäyksiltä, koska niiden toteutus aiheuttaa rikollisille niin paljon vaivaa suhteessa hyötyyn. Tosiasiassa kuitenkin teknologia ja laskennallinen teho kehittyvät jatkuvasti eteenpäin, kyberrikollisuus pysyy tuotollisena, ja ehkä jo viiden vuoden päästä laajoihin botnet-verkkoihin saadaan automatisoitua väistötekniikat, joilla kierretään kotien tavalliset palomuurit. Väittäisin, että IDS tuo vähintäänkin mielenrauhaa.

Työn teki jokseenkin haasteelliseksi suhteellisen vähä kokemus Linux-ympäristöjen peruskäytöstä – tai ainakin se hidasti prosessia – ja Snortin DAQ-moduuli. Alun perin rakensin Snort-palomuurin kahden verkon välissä reitittävään virtuaalilaitteeseen, ja mahdollisesti suurin osa koko projektista kului vianetsintään, kun Snort ei jostain syystä saanut muokattua verkkoliikennettä eivätkä lukuisat dokumentaatiot auttaneet. Sain siis aivan itse keksiä, että DAQ-moduuli afpacket on tarkoitettu vain verkkoliikennettä kytkevälle laitteelle eikä reitittävälle.

Sen lisäksi, että tämä työ antoi minulle hyvän perusymmärryksen väistötekniikoista, kartutin myös mukavasti omia Linuxin perustaitojani. Myös verkkoliikenteen tuntemukseni syveni merkittävästi, vaikkakin lähinnä kohdista, jotka liittyvät olennaisesti väistötekniikoihin.

Lähteet

1 NSS Labs Tests Show Next Generation Firewall Security Effectiveness Remains Strong. 2014. Verkkodokumentti. NSS Labs. <https://www.nsslabs.com/news/press-releases/nss-labs-tests-show-next-generation-firewall-security-effectiveness-remains>. Luettu 1.4.2015.

2 Pfleeger, Charles P. & Pfleeger, Shari Lawrence. 2009. Security in Computing. Fourth Edition. New Jersey: Prentice Hall.

3 Dostálek, Libor & Kabelová, Alena. 2006. Understanding TCP/IP. Birmingham, UK: Packt Publishing.

4 Odom, Wendell. 2011. The TCP/IP and OSI Networking Models. Verkkodokumentti. Cisco Press. <http://www.ciscopress.com/articles/article.asp?p=1757634&seqNum=2>. Luettu 22.4.2015.

5 Transmission Control Protocol. 1981. Verkkodokumentti. IETF. <https://tools.ietf.org/html/rfc793>. Luettu 19.4.2017.

6 Braden, R. 1989. Requirements for Internet Hosts -- Communication Layers. Verkkodokumentti. IETF. <https://tools.ietf.org/html/rfc1122>. Luettu 19.4.2017.

7 Gont, F. & Yourtchenko, A. 2011. On the Implementation of the TCP Urgent Mechanism. Verkkodokumentti. IETF. <https://tools.ietf.org/html/rfc6093>. Luettu 19.4.2017.

8 Internet Protocol. 1981. Verkkodokumentti. IETF. <https://tools.ietf.org/html/rfc791>. Luettu 19.4.2017.

9 Beaume, Jeremy. 2017. Suricata IDS 3.2.12 IPv4 evasion. Verkkodokumentti. CXSecurity. <https://cxsecurity.com/issue/WLB-2017020165>. Luettu 19.4.2017.

10 Tsung-Huan, Cheng; Ying-Dar, Lin; Yuan-Cheng, Lai & Po-Ching, Lin. 2011. Evasion Techniques: Sneaking through Your Intrusion Detection/Prevention Systems. IEEE.

11 García-Teodoro, P.; Díaz-Verdejo, J.; Maciá-Fernández, G. & Vázquez, E. 2008. Anomaly-based network intrusion detection: Techniques, systems and challenges. Elsevier.

12 Esler, Joel. 2009. Why is your IDS outside your firewall? Verkkodokumentti. <http://blog.joelesler.net/2009/03/why-is-your-ids-outside-your-firewall_06.html>. Luettu 18.4.2017.

13 Pfeifer, Bill. 2011. IPS vs. Deep packet inspection. Verkkodokumentti. Juniper. <https://forums.juniper.net/t5/SRX-Services-Gateway/IPS-vs-Deep-packet-inspection/td-p/108674>. Luettu 18.4.2017.

14 Siby, Subin. 2016. Default TTL (Time To Live) Values of Different OS. Verkkodokumentti. <http://subinsb.com/default-device-ttl-values>. Luettu 21.4.2017.

15 Ko, Youngjun. 2001. NIDS Evasion Method named "SeolMa". Verkkodokumentti. Phrack Magazine. <http://phrack.org/issues/57/3.html>. Luettu 1.4.2017.

16 Andersson, Stig; Clark, Andrew & Mohay, George M. 2005. Detecting Network-based Obfuscated Code Injection Attacks Using Sandboxing. Queensland University of Technology.

17 Ptacek, Thomas H. & Newsham, Timothy N. 1998. Insertion, evasion, and denial of service: Eluding network intrusion detection. Verkkodokumentti. Secure Networks, Inc. <http://insecure.org/stf/secnet_ids/secnet_ids.html>. Luettu 18.4.2017.

18 Day, David J. & Burns, Benjamin M. 2011. A Performance Analysis of Snort and Suricata Network Intrusion Detection and Prevention Engines. Verkkodokumentti. IARIA. <http://www.thinkmind.org/download.php?articleid=icds_2011_7_40_90007>. Luettu 1.4.2015.

19 Niemi, Olli-Pekka. 2012. Multilayer Fuzzing With Evader. Verkkodokumentti. DeepSec. <https://deepsec.net/docs/Slides/2012/DeepSec_2012_Olli-Pekka_Niemi_-_Multilayer_Evasion_Fuzzing_with_Evader.pdf>. Luettu 18.4.2017.

20 Snort. 2017. Verkkodokumentti. <https://snort.org>. Luettu 13.4.2017.